Math 405 Assignment 4

Kaan Yolsever

November 2018

1. The h values and errors are given together:

   [0.25, 3.64643971413e-05], [0.125, 1.20255837825e-05], [0.0625, 3.21031885488e-06], [0.03125, 8.15651403402e-07]

   And my code for this part is:

```
A = [4 ,8, 16, 32]
for N in A:
    h = 1/N
    x = linspace(0,1-h,N)
    u0 = sin(2*pi*x)

    L = np.zeros((N,N))

    for r in range(N):
        for c in range(N):
            if(r==c):
                L[r,c] = -2
            elif (c == r+1):
                L[r,c] = 1
            elif (c == r-1):
                L[r,c] = 1
    L[0,N-1] = 1
    L[N-1,0] = 1
    L = 1/h**2*L;

    Tf = 0.25;
    k = 0.25*h**2;
    numsteps = ceil(Tf/k)
    k = Tf/numsteps
    numsteps = int(numsteps)
    u = u0;
    for n in range(numsteps):
        unew = u + k*(np.dot(L,u));
        u = unew;
    u_exact = e**(-4*pi**2*Tf)*sin(2*pi*x);
    error = max(abs(u-u_exact))
    h_values = []
    error_values = []
    h_values = [h_values, h]
    error_values = [error_values, error]
    print(h_values)
    print(error_values)
```

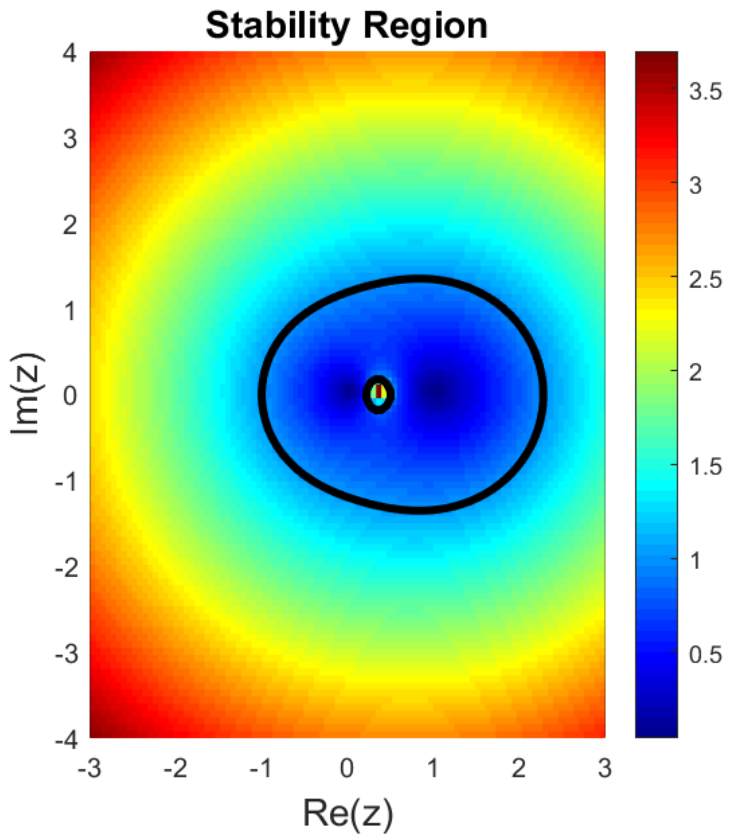
2. (a) Assume f=0 and u(0)= 0 $u^{n+1} = u^n$
   and with ansatz $u^{n+1} = g^{n+1}$

and with telescoping: we see $g^n = g^{n-1}$ and so on then $g^{n+1} = 1$ thus the method is zero-stable

(b) $u^{n+1} - u^n = k(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}$ apply $f(t,u^n) = \lambda u^n$ and set $u^n = z^n$ then we get $\lambda k = \frac{z^2 - z}{\frac{3}{2}z - \frac{1}{2}}$

Plotting this region with matlab:

(c) The absolute stability region looks like this:



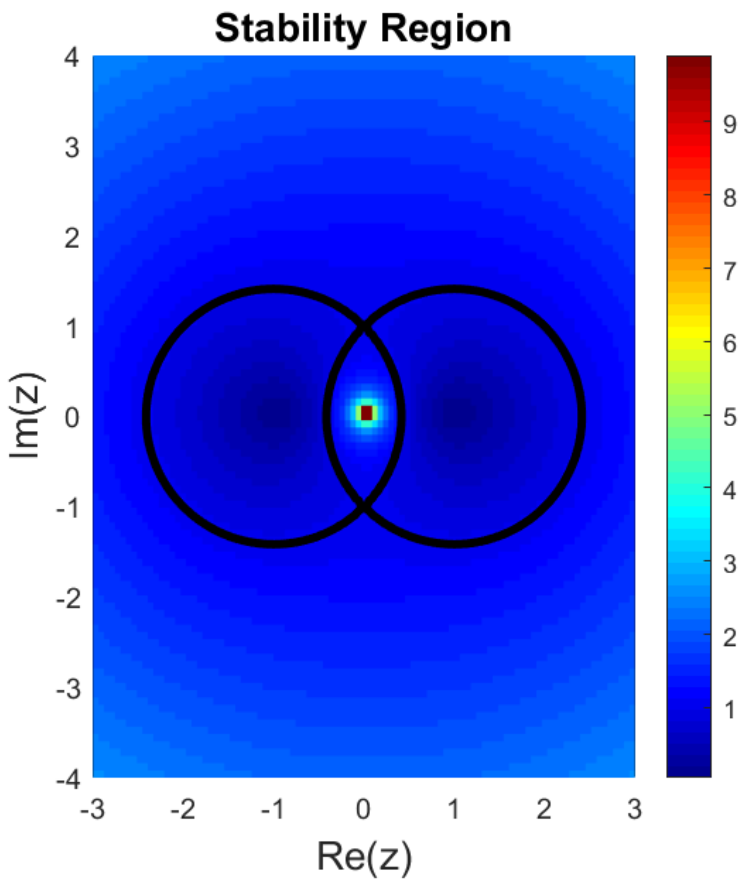
My code for this part is:

```
z_re1d = linspace(-3,3,100);
z_im1d = linspace(-4,4,100);
[z_re, z_im] = meshgrid(z_re1d, z_im1d);
z = z_re + 1i*z_im;
G = (z.^2 - z)./(3/2*z - 1/2);
figure(1), clf
pcolor(z_re, z_im, abs(G)); shading flat; hold on;
contour(z_re, z_im, abs(G), [1 1], 'k', 'linewidth', 3);
grid on, axis equal;
xlabel('Re(z)','fontsize',14);
ylabel('Im(z)','fontsize',14);
title('Stability Region', 'fontsize', 14);
colorbar
colormap jet
```

(d) As k goes to 0, our method is going to behaves like $U^{n+1} - U^n = 0$. Since the right hand side in this case is 0 just like when it was when we were testing for zero stability, so the stability in this case is the same as (6a)

3. $f(u^n) = \frac{u^{n+1}-u^{n-1}}{2k}$

Using the same technique as question 2c, $f(u^n) = \lambda u^n$ and setting $u^n = g^n$ we get $\lambda k = \frac{g^2}{2g}$ and plotting this via matlab:



In all the other methods, we had non-trivial absolute stability regions but we do not with this method.

My code for this part:

```
z_re1d = linspace(-3,3,100);
z_im1d = linspace(-4,4,100);
[z_re, z_im] = meshgrid(z_re1d, z_im1d);
z = z_re + 1i*z_im;
G = (z.^2 - 1)./ (2*z);
figure(1), clf
pcolor(z_re, z_im, abs(G)); shading flat; hold on;
contour(z_re, z_im, abs(G), [1 1], 'k', 'linewidth', 3);
grid on, axis equal;
xlabel('Re(z)','fontsize',14);
ylabel('Im(z)','fontsize',14);
title('Stability Region', 'fontsize', 14);
colorbar
colormap jet
```

4. $\frac{u_j^{n+1}-u_j^{n-1}}{2k} + a\frac{u_{j+1}^n-u_{j-1}^n}{2h} = 0$

plug in $u_j^n = e^{i\xi hj}$ and put everything on common denominators then we get:

$g^2(\xi)e^{ijh\xi} - e^{ijh\xi} + \nu g e^{i\xi(j+1)h} - v g e^{i\xi(j-1)h} = 0$

diving by $e^{ijh\xi}$ and substituting $e^{ix} = cosx + isinx$ we get:

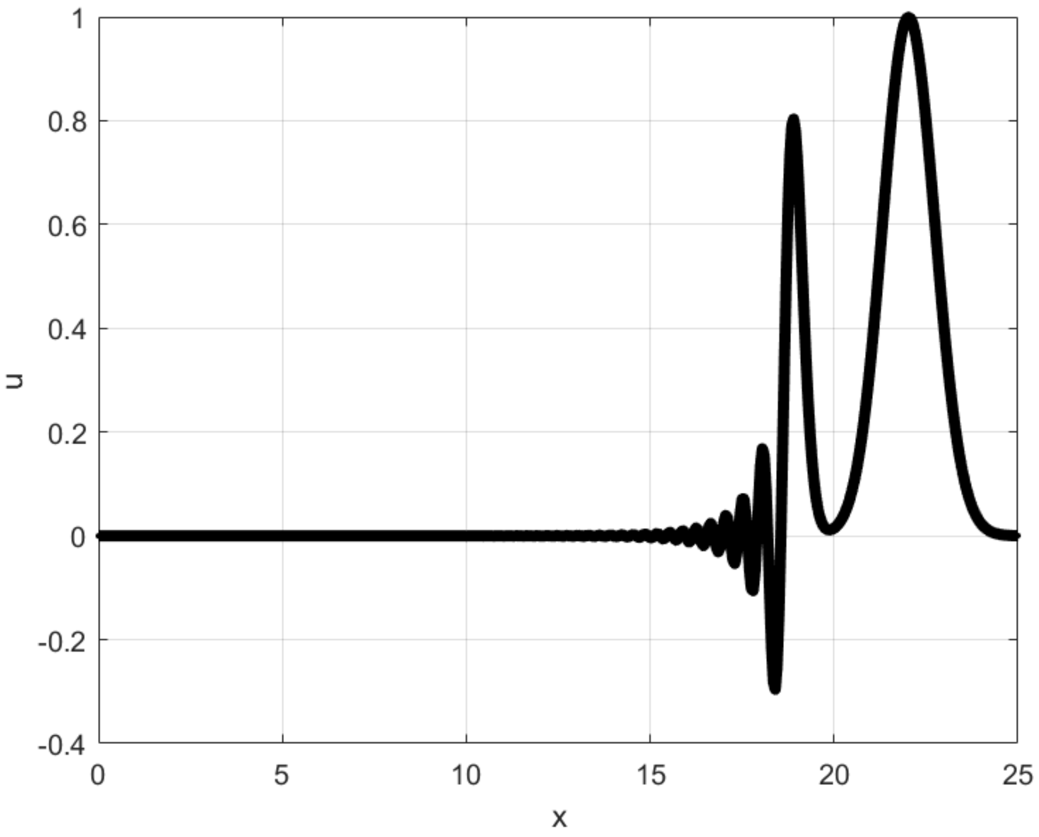
$g^2 + 2\nu g(isin(\xi h)) - 1 = 0$

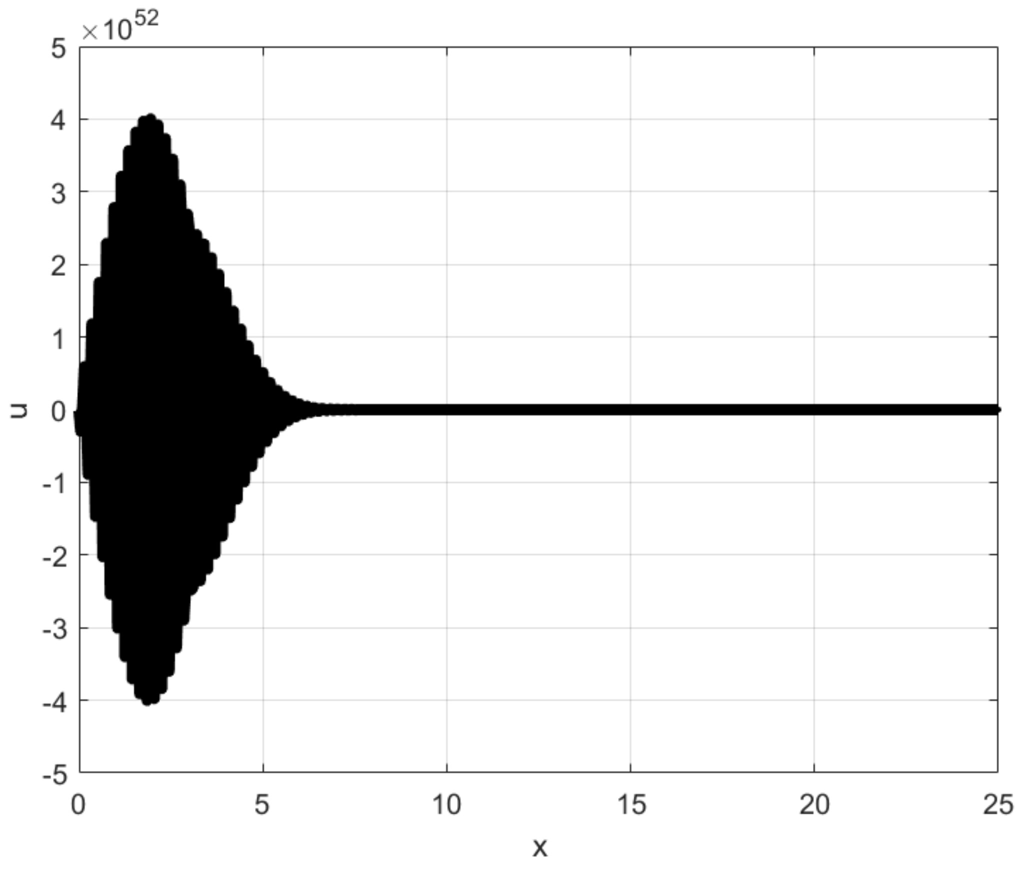
Using the discriminant

$\Delta = 4\nu^2(i^2 sin^2(\xi h)) + 4 = -4v^2 sin^2(\xi h) + 4$

Then we can see that for any $\nu^2 sin^2(\xi h) < 1$ implies that $g < 1$, thus this 2^{nd} order method works perfectly: the amplitude never grows or decays. since we need $\nu^2 sin^2(\xi h) < 1$ and $sin^2 < 1$ this implies $\nu^2 < 1$ and this implies $k^2 < \frac{h^2}{a^2}$

5. In parallel with my finding from the last question the solution blows up when $\nu = 1.1, h = 0.05$ and it does not blow up when $\nu = 0.8, h = 0.05$ as shown in the diagrams below(the first one is for 0.8):



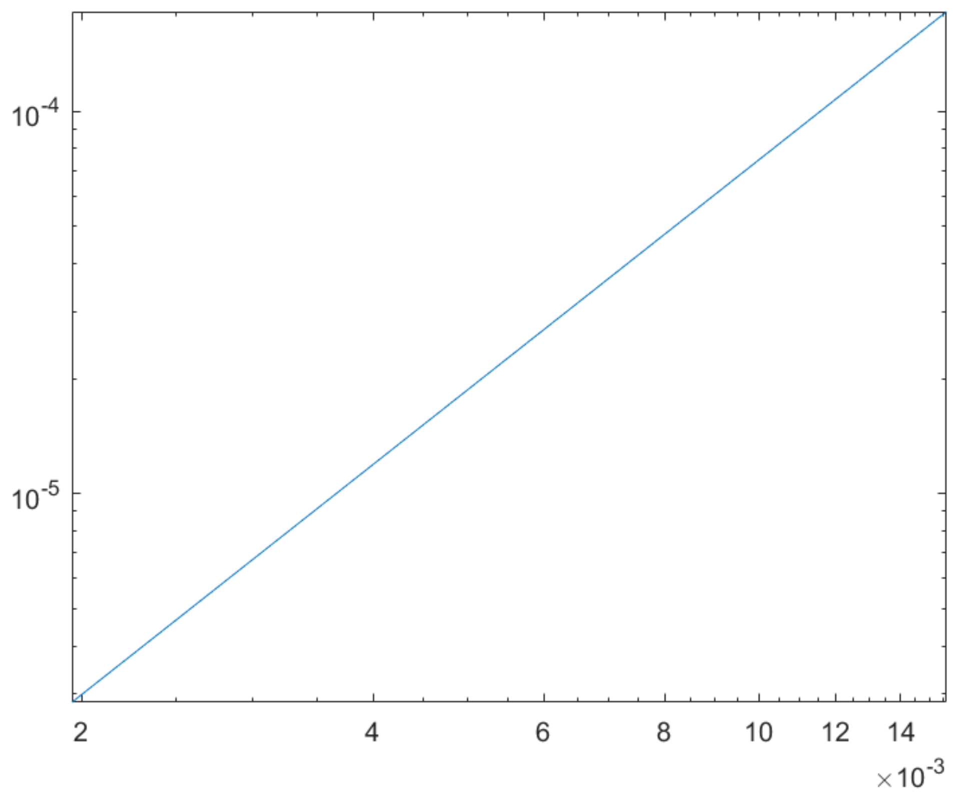


My code for this part is:

```
% Grid and initial data:
h = .05;                    % space step
k = 0.8*h;                  % time step
x = (0:h:25)';           % grid
v0 = exp(-20*(x-2).^2) + exp(-(x - 5).^2);
v1 = exp(-20*(x-k-2).^2) + exp(-(x-k-5).^2);
N = length(x);
Tf = 17;

figure(1); clf;
plt = plot(x, v0, 'k.-', 'linewidth', 4);
%axis([-1 1 -.01 1.01])
grid on
xlabel('x');  ylabel('u');

% adjust either final time or time-step to have integer steps
numsteps = ceil(Tf / k);
k = Tf / numsteps;

% Time-stepping:
for n=1:numsteps
  % loop over spatial points
  for j=1:N
    if n == 1
       if j == 1
      % special case for Left BC
```

```
       vnew(j) = - v1(j+1) / h * k + v0(j);
       elseif j == N
      % special case for Right BC
       vnew(j) = v1(j-1) / h * k + v0(j);
       else
       vnew(j) = (v1(j-1) - v1(j+1)) / h * k + v0(j);
       end
    else
        if j == 1
      % special case for Left BC
       vnew(j) = - v(j+1) / h * k + vold(j);
       elseif j == N
      % special case for Right BC
       vnew(j) = v(j-1) / h * k + vold(j);
       else
       vnew(j) = (v(j-1) - v(j+1)) / h * k + vold(j);
       end
       end
  end
  if n== 1
     v = vnew;
     vold = v1;
     set(plt, 'ydata', v)
     drawnow
  end
  if n ~= 1
     vold= v;
     v = vnew;
     set(plt, 'ydata', v )
     drawnow
  end
end
```

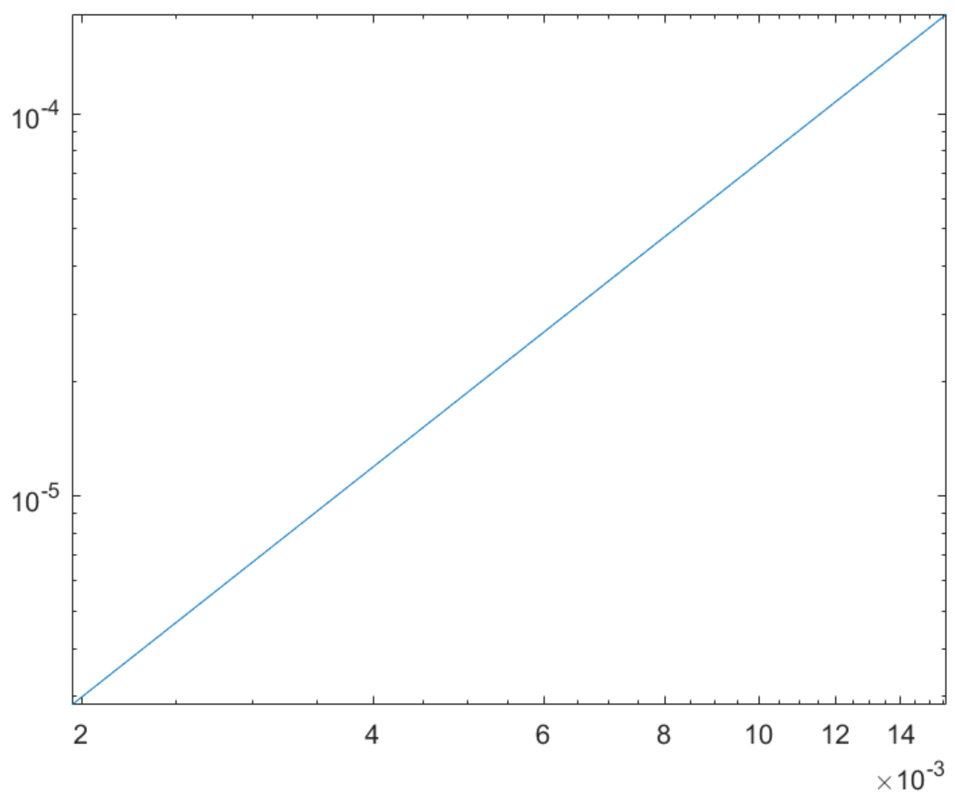
6. (a) The results of my convergence study where y is the log error and x is h values:



and my code:

```
clear all
clc
h_results = [];
error_results = [];
A = [1/64 1/128 1/256 1/512];
for h = A
x = (h:h:(1-h))';   % interior pts
m = length(x);
e = ones(1, m)';
Dxx = spdiags([e -2*e e], [-1 0 1], m, m);
%full(Dxx)
Dxx = Dxx/h^2;
% RHS function
f = (-1-pi^2)*sin(pi*x);
% exact soln for comparing/plotting
u_exact = sin(pi*x);
% solve Dxx*U = F for U
u_approx = (Dxx-eye(m)) \ f;
h_results = [h_results h];
error_results = [error_results norm(u_approx - u_exact, inf)];
end
loglog(h_results,error_results)
```

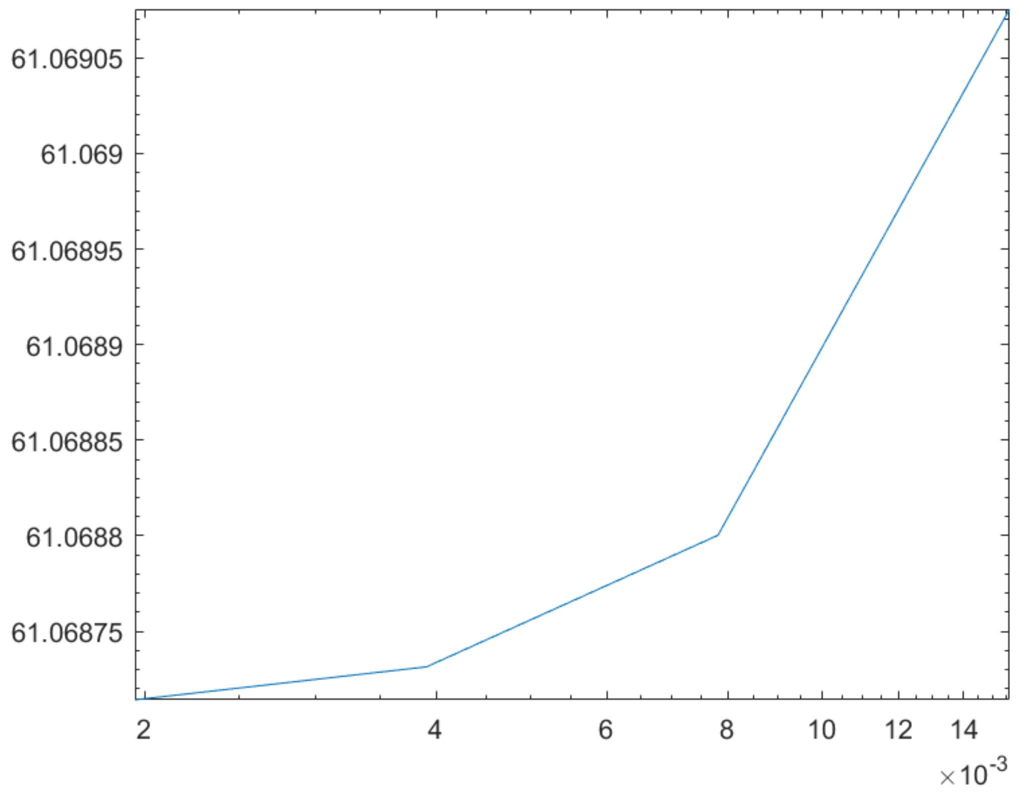
(b) The error-h graph looks like:



My code for this part:

```
clear all
clc
h_results = [];
error_results = [];
A = [1/64 1/128 1/256 1/512];
for h = A
%x = (h:h:(1-h))';   % interior pts
N = h^(-1);
x = linspace(0, 1, N);
e = ones(1, N)';
Dxx = spdiags([e -2*e e], [-1 0 1], N, N);
%full(Dxx)
Dxx = Dxx/h^2;
% RHS function
f = 3.^x*(log(3)*log(3)-1);
f(1) = f(1) -1/h^2;
f(N) = f(N) -3/h^2;
f = f';
% exact soln for comparing/plotting
u_exact = 3.^x;
% solve Dxx*U = F for U
u_approx = (Dxx-eye(N)) \ f;
h_results = [h_results h];
error_results = [error_results norm(u_approx - u_exact, inf)];
end
loglog(h_results,error_results)
```

(c) The error-h plot is:



My code for this part:

```
h_results = [];
error_results = [];
A = [1/64 1/128 1/256 1/512];
for h = A
x = (0:h:(1-h))';   % interior pts
m = length(x);
e = ones(1, m)';
Dxx = spdiags([e -2*e e], [-1 0 1], m, m);
Dxx(1,m) = 1;
Dxx(m,1) = 1;
Dxx = Dxx/h^2;
f = -4*pi^2*e.^(sin(2*pi*x)).*(sin(2*pi*x)-cos(2*x*pi).*cos(2*pi*x)-1);
u_exact = e.^sin(2*pi*x);
u_approx = (Dxx-eye(m)) \ f;
h_results = [h_results h];
error_results = [error_results norm(u_approx - u_exact, inf)];
end
loglog(h_results,error_results)
```

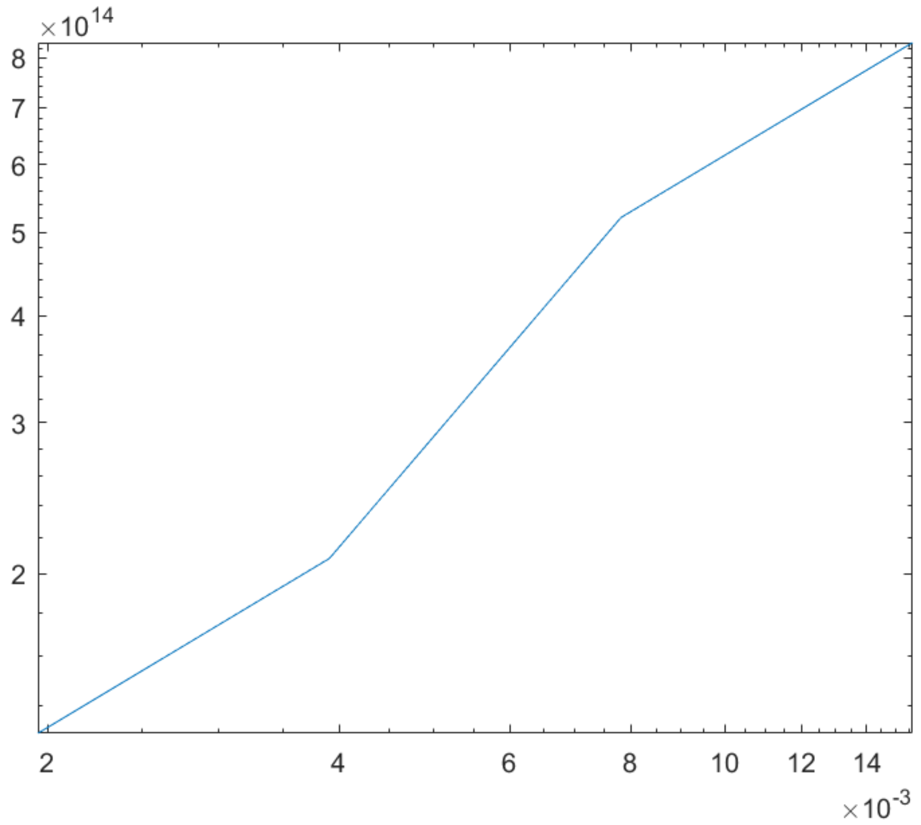
(d) my code for this part:

```
clear all
clc
h_results = [];
error_results = [];
A = [1/64 1/128 1/256 1/512];
for h = A
```

```
x = (h:h:(1-h))';   % interior pts

m = length(x);

e = ones(1, m)';
Dxx = spdiags([e -2*e e], [-1 0 1], m, m);
Dxx = Dxx/h^2;
% RHS function
f = (-1-pi^2)*cos(pi*x);
% exact soln for comparing/plotting
u_exact = cos(pi*x);
% solve Dxx*U = F for U
u_approx = (Dxx-eye(m)) \ f;
h_results = [h_results h];
error_results = [error_results norm(u_approx - u_exact, inf)];
end
loglog(h_results,error_results)
```

(e) I get an error which says the matrix is singular. Thus the condition number is infinite and the solution is not unique. Since condition number is 0, the function is very sensitive to small perturbations. Thus the errors blow up.

The error-h plot is:



7. (a) Guess u(x,y) = Af(x,y) then $-u_{xx} - u_{yy} = 5A\pi^2 f(x,y) = f(x,y)$ which implies A = $\frac{1}{5\pi^2}$ Thus u(x,y) = $\frac{1}{5\pi^2} sin(\pi x) sin(2\pi y)$

And the related relative errors: 0.0027, 6.8297e-04,1.7069e-04,4.2670e-05 so the error behaves like $O(h^2)$

¿¿

My code for this part:

```
%% 2D Laplacian example with non-homogeneous boundary conditions

A = [32 64 128 256];
for N = A
hx = 1/N;
hy = 1/N;
x1d = 0:hx:2;
y1d = 0:hy:1;

[xx, yy] = meshgrid(x1d, y1d);
x = xx(:);
y = yy(:);

%% "method of manufactured solutions"
% Choose $u(x,y)$ then sub into PDE to find $f$
uexact_fcn = @(x, y) 1/(5*pi^2).*sin(pi.*x).*sin(2*pi.*y);
f_fcn = @(x, y) sin(pi.*x).*sin(2*pi.*y);
g = @(x,y) 0;

f = f_fcn(x, y);
uexact = uexact_fcn(x, y);

%% index sets for the boundaries
% note these operlap at the corners
b1 = (x == x1d(1));
b2 = (x == x1d(end));
b3 = (y == y1d(1));
b4 = (y == y1d(end));

%% Boundary condition vector for non-homogenous part
bc_rhs = zeros(size(x));
bc_rhs(b1) = bc_rhs(b1) + 2/hx^2*g(x(b1), y(b1));
bc_rhs(b2) = bc_rhs(b2) + 2/hx^2*g(x(b2), y(b2));
bc_rhs(b3) = bc_rhs(b3) + 2/hy^2*g(x(b3), y(b3));
bc_rhs(b4) = bc_rhs(b4) + 2/hy^2*g(x(b4), y(b4));

%% Right-hand side is f + boundary data
rhs = f + bc_rhs;

%% Sparse matrix to execute finite difference operation:
[Dxx, Dyy, Dxc, Dyc, Dxb, Dyb, Dxf, Dyf, Dxyc] = ...
  diff2d_matrices(x1d, y1d, 0, 'd');
L = Dxx + Dyy;

%% Solve
u = -L \ rhs;

%% Plots and error

rel_abs_err = norm(uexact - u, inf) / norm(uexact, inf)
end
```

(b) Just changing the domain to x1d = 0:hx:(2-h) in the previous code. The relative errors become:

0.0957, 0.0488, 0.0245,0.0123. So, the error behaves like O(h)

(c) Using matlab's double integral calculator we find that a= 31.8310
My code for this part is:

```
fun = @(x,y) exp(-((x-1.5).^2+(y-0.6).^2)*100);
q = integral2(fun,0,2,0,1);
a = 1/q;
```

(d)